Universally Composable Adaptive Oblivious Transfer

Matthew Green Susan Hohenberger

Information Security Institute
The Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218
{mgreen,susan}@cs.jhu.edu

Abstract

In an oblivious transfer (OT) protocol, a Sender with messages $M_1, \ldots, M_N$ and a Receiver with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the Receiver obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages, and the Sender does not learn anything about $\sigma_1, \ldots, \sigma_k$. In an *adaptive* protocol, the Receiver may obtain $M_{\sigma_{i-1}}$ before deciding on σ_i. Efficient adaptive OT protocols are interesting both as a building block for secure multiparty computation and for enabling oblivious searches on medical and patent databases.

Historically, adaptive OT protocols were analyzed with respect to a "half-simulation" definition which Naor and Pinkas showed to be flawed. In 2007, Camenisch, Neven, and shelat, and subsequent other works, demonstrated efficient adaptive protocols in the full-simulation model. These protocols, however, all use standard rewinding techniques in their proofs of security and thus are not universally composable. Recently, Peikert, Vaikuntanathan and Waters presented universally composable (UC) *non-adaptive* OT protocols (for the 1-out-of-2 variant). However, it is not clear how to preserve UC security while extending these protocols to the adaptive k-out-of-N setting. Further, any such attempt would seem to require $O(N)$ computation per transfer for a database of size N. In this work, we present an efficient and UC-secure *adaptive* k-out-of-N OT protocol, where after an initial commitment to the database, the cost of each transfer is *constant*. Our construction is secure under bilinear assumptions in the standard model.

1 Introduction

Oblivious transfer (OT) was introduced by Rabin [29] and generalized by Even, Goldreich and Lempel [19] and Brassard, Crépeau and Robert [9]. It is a two-party protocol, where a Sender with messages $M_1, \ldots, M_N$ and a Receiver with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the Receiver obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages and the Sender does not learn anything about $\sigma_1, \ldots, \sigma_k$. Naor and Pinkas were the first to consider an *adaptive* setting, $\mathsf{OT}^N_{k\times 1}$, where the Receiver may obtain $M_{\sigma_{i-1}}$ before deciding on σ_i [26]. Efficient OT schemes are very important. OT^4_1 is a key building block for secure multi-party computation [31, 20, 24]. $\mathsf{OT}^N_{k\times 1}$ is a useful and interesting tool in its own right, enabling oblivious databases for applications such as medical record storage and patent searches [27].

Unfortunately, developing efficient *adaptive* protocols appears to be a more difficult and involved process than that of developing efficient *non-adaptive* protocols. Indeed, even finding the right security definition has proven challenging. Historically, many efficient OT constructions were analyzed under a "half-simulation" definition, where the Sender and Receiver's security are described by a combination of simulation and game-based definitions. Naor and Pinkas [26] showed that schemes analyzed under this definition may admit practical attacks on the Receiver's privacy. To address this, Camenisch, Neven and shelat [10] and subsequently Green and Hohenberger [21] proposed efficient and fully-simulatable $\mathsf{OT}^N_{k\times 1}$ protocols under bilinear assumptions. Each of these protocols achieve the optimal total communications cost of $O(N + k)$ with reasonable constants. Unfortunately, the security proofs for these protocols employ adversarial rewinding, and thus do not imply security under concurrent execution.

Recently, Lindell [25] showed how to achieve efficient and fully-simulatable *non-adaptive* OT^2_1 under the DDH, Nth residuosity and quadratic residuosity assumptions, as well as the assumption that homomorphic encryption exists. Simultaneously, Peikert, Vaikuntanathan and Waters [28] proposed several non-adaptive, but universally composable OT^2_1 protocols based on DDH, quadratic residuosity and lattice-based assumptions. While both of these valuable works add to our collective knowledge for non-adaptive OT, they do not shed much light on how to achieve efficient adaptive protocols. Indeed, Lindell points out that the adaptive case is considerably harder [25].

The general framework used in [25, 28] (where the Receiver chooses the encryption keys) seems inherently at odds with allowing efficient adaptive schemes. Each transfer requires $O(N)$ work for the Sender, whereas this can be *constant* in our protocols. Even more alarming, it isn't clear how (without killing the efficiency and perhaps the UC security of [28]) a Sender could convince the Receiver that he is not changing the database values with each request. This problem of ensuring a *consistent* database gets even worse when multiple Receivers are considered, as we do in Section 5.

Our Results. In this work, we take a different approach to constructing OT protocols, which allows them to be simultaneously efficient, adaptive, universally composable and globally consistent. We summarize what is known about $\mathsf{OT}^N_{k\times 1}$ protocols in Figure 1. Let us describe some highlights.

1. *Universal Composability:* The Universal Composability framework [13] allows for the design of concurrent and composable cryptographic protocols, which are important properties in any practical deployment of an oblivious database. Canetti and Fischlin showed that OT cannot be UC-realized without additional trusted setup assumptions such as the existence of a Common Reference String (CRS) [15]. This is formally referred to as the $\mathcal{F}_{CRS}$-hybrid model, and is assumed by the constructions of Peikert *et al.* [28] as well as those in this work.

2. *Efficiency:* Our protocol is practical. The initialization phase requires $O(N)$ communication cost, and each transfer phase requires only constant cost, for reasonable constants. Our CRS can be only 15 group elements, and can be sampled from an arbitrary common random string (see Sections 4.1 and 4.2 for more details.) In contrast, simply repeating a OT^N_1 scheme (such as [28]) k times would require $O(N)$ communication cost for *each* transfer plus the additional work required for the Sender to convince the Receiver that he isn't changing the database values dynamically.

   Moreover, the message space of our protocol is a group element (so at least 160 bits), whereas the quadratic residuosity and lattice-based schemes of [28] have *one*-bit message spaces. We note, however, that the DDH-based scheme of [28] allows for longer messages.

3. *Global Consistency:* In our constructions, the sender publishes some form of commitment to the database at the beginning of the protocol. When joint state is allowed (see Sections 2

Protocol	Rounds	Communication	Assumption
Half Simulation:			
NP99 [26]	$\ell k \, log \, N + 1/2$	–	Sum Consistent Synthesizers + ℓ-round OT_1^2
CT05 [18]	$O(k) + 1/2$	$O(N)$	Decisional DH (in ROM)
Full Simulation:			
CNS07 [10]	$4k + 1/2$	$O(N)$	y-Power Decisional DH + q-Strong DH
CNS07 [10]	$O(k) + 1/2$	$O(N)$	Unique blind signature (in ROM)
GH07 [21]	$k + 1/2$	$O(N)$	Decisional Bilinear DH (in ROM)
UC ($\mathcal{F}_{CRS}$-hybrid):			
This work (§4)	$k + 1/2$	$O(N)$	q-Strong Decision Linear + Uniform q-Hidden Strong DH

Figure 1: Survey of efficient, adaptive k-out-of-N Oblivious Transfer protocols.

and 5), then multiple receivers, all of whom start with the same commitment, can be sure that whenever their request on index i succeeds, they receive the same message M_i as any other receiver. In other words, globally consistency ensures that a sender cannot return patent M_i to Alice and a different patent $M_i' \neq M_i$ to Bob.

4. *Model and Assumptions:* Our construction does not require random oracles and can be implemented under Uniform q-Hidden Strong Diffie-Hellman [8, 3] and q-Strong Decision Linear.

Intuition behind the Construction. Oblivious Transfer protocols can be roughly divided into two categories. Let's restrict our attention to non-adaptive OT_1^N for the moment. In approach (1), which is used by [29, 19, 25, 28], the Receiver transmits a collection of specially-formed encryption keys to the Sender, who encrypts each message and returns the N ciphertexts to the Receiver. The protocol is secure provided that the encryption keys are formed such that a Receiver is able to decrypt at most *one* of the resulting ciphertexts. In approach (2), which is used by [10, 21] and this work, the Sender encrypts the message collection under keys of her own choosing, and— in some interactive protocol with the Receiver— helps to decrypt *one* ciphertext.

While both approaches can be used to implement adaptive OT, the first approach requires that the Sender generate a new set of ciphertexts at *each* transfer stage (for *each* receiver), requiring at least $O(N \cdot k)$ cost. Even worse, the Sender might be able to maliciously change the database from one transfer stage to another and to present different versions of the database to different receivers.

The latter approach is better suited for the adaptive case. A single database can be committed to and then each decryption can be performed in constant computational and communication cost, for a total $O(N + k)$ cost. This approach is taken by the fully-simulatable protocols of [10, 21], which both use rewinding in their simulations to (1) simulate proofs and (2) extract knowledge. Our protocol in Section 4 achieves composability by using the proof techniques of Groth and Sahai [23]. This is non-trivial for three reasons. First, the Groth-Sahai proofs have not yet been shown to be either simulation-sound or UC in general, and thus we must be careful how we use them. Second, the Groth-Sahai techniques provide broad support for non-interactive, *witness indistinguishable* proofs of algebraic assertions in bilinear groups, but only provide non-interactive, *zero-knowledge* proofs for a restricted class of algebraic assertions. This requires us to design an efficient OT protocol from a more restricted set of NI tools. Third, all known security proofs of adaptive OT protocols require some form of extraction (e.g., extracting the chosen index from the adversarial Receiver or extracting the secret encryption keys from the adversarial Sender); unfortunately, Groth-Sahai proofs of knowledge are f-extractable (but not fully extractable), where only some one-way function of the witness, $f(w)$, can be extracted and not the witness w itself.[1] This complicates matters. Thus, while these new proof techniques are useful and powerful, they cannot simply be plugged into existing adaptive OT protocols. Indeed, our construction starts from scratch.

2 Definitions

Notation. By OT_k^N (resp., $\mathsf{OT}_{k\times 1}^N$), we denote a non-adaptive (resp., adaptive) k-out-of-N oblivious transfer protocol. Let $\overset{c}{\approx}$ denote computational indistinguishability, as defined in [13].

Adaptive k-out-of-N Oblivious Transfer. $\mathsf{OT}_{k\times 1}^N$ protocols consist of two phases: Initialization and Transfer. In the Initialization phase, the Sender commits to the input database $M_1, \ldots, M_N$. Subsequently, the Sender and Receiver engage in up to k Transfers. During the i^{th} Transfer, the Receiver adaptively selects a message index $\sigma_i \in [1, N]$ and engages in a protocol such that it obtains M_{σ_i} (or $\perp$ if the protocol fails) and nothing else, while the Sender learns nothing about σ_i. The simulation-based nature of the security definition we use ensures that protocol failures must occur independently of the message index σ_i chosen by the Receiver (capturing the strong selective-failure blindness property [10].)

Universally Composable Security. Here, as in [28], we work in the standard UC framework with static corruptions, where all parties are modeled as p.p.t. interactive Turing machines. Security of protocols is defined by comparing the protocol execution to an *ideal process* for carrying out the desired task. More formally, there is an *environment* $\mathcal{Z}$ whose task is to distinguish between two worlds: ideal and real. In the ideal world, "dummy parties" (some of whom may be corrupted by the *ideal adversary* $\mathcal{S}$) interact with an *ideal functionality* $\mathcal{F}$. In the real world, parties (some of whom may be corrupted by the *real world adversary* $\mathcal{A}$) interact with each other according to some protocol π. We refer to Canetti [13, 14] for a fuller description, as well as a definition of the ideal world ensemble $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and the real world ensemble $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. We use the established notion of a protocol π *securely realizing* an ideal functionality $\mathcal{F}$ as:

Definition 2.1 *Let $\mathcal{F}$ be a functionality. A protocol π is said to UC-realize $\mathcal{F}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$,*

$$\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}.$$

Canetti and Fischlin showed that OT cannot be UC-realized without some form of trusted setup assumption [15]. Thus, as in [16, 28], we assume the existence of an honestly-generated Common Reference String (crs), and are thus secure in the so-called $\mathcal{F}_{CRS}$-hybrid model. The functionality is parameterized by a distribution D and a set $\mathcal{P}$ of recipients. For our purposes, $\mathcal{P}$ will include the OT Sender and Receiver only. Here the environment $\mathcal{Z}$ learns about the reference string value from the adversary, and thus the simulator can set up a reference string with "trapdoor information" or even with a slightly different distribution.

Figure 2 describes the $\mathcal{F}_{CRS}$ functionality and Figure 3 describes the $\mathcal{F}_{OT}^{N\times 1}$ functionality.

We briefly mention that there are techniques for designing and analyzing multiple OT protocols which use a single reference string; i.e., a multi-session extension. One might worry that if multiple

[1]For example, if the witness is a value x, the extractor might be able to recover the value g^x.

Functionality $\mathcal{F}_{CRS}^{\mathcal{D},P}$

Upon receiving input $(\mathsf{sid}, \mathsf{crs})$ from party P, first verify that $p \in \mathcal{P}$; else ignore the input. If there is no value r recorded, then choose and record $r \leftarrow D$. Finally send output $(\mathsf{sid}, \mathsf{crs}, r)$ to P.

Figure 2: Ideal functionality for the common reference string [14].

Functionality $\mathcal{F}_{OT}^{N\times 1}$

$\mathcal{F}_{OT}^{N\times 1}$ proceeds as follows, parameterized with integers N, ℓ and running with an oblivious transfer Sender **S**, a receiver **R** and an adversary $\mathcal{S}$.

- Upon receiving a message $(\mathsf{sid}, \mathsf{sender}, m_1, \ldots, m_N)$ from **S**, where each $m_i \in \{0,1\}^\ell$, store $(m_1, \ldots, m_N)$.
- Upon receiving a message $(\mathsf{sid}, \mathsf{receiver}, \sigma)$ from **R**, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously received. If no such message was received, send nothing to **R**. Otherwise, send $(\mathsf{sid}, \mathsf{request})$ to **S** and receive the tuple $(\mathsf{sid}, b \in \{0,1\})$ in response. Pass (sid, b) to the adversary, and: If $b = 0$, send $(\mathsf{sid}, \perp)$ to **R**. If $b = 1$, send (sid, m_σ) to **R**.

Figure 3: Functionality for adaptive Oblivious Transfer, based on the OT_1^2 definition from [16].

protocols now share some joint state, then they can no longer be analyzed separately and then composed later. Fortunately, this is addressed by *universal composition with joint state* (JUC) [17] and could be done in our case. A second issue with sharing the reference string is that we make no guarantee about the security of protocols which use the same reference string in ways other than those specified by the OT protocol, and here we explicitly assume that the crs is only available to certain parties. This is at odds with the notion that the crs is a "global" entity, however, there are strong impossibility results for UC-realizing OT in a setting where the crs is available to everyone (including the environment) and can no longer be crafted by the simulator. There are models, such as the *augmented CRS* functionality $\mathcal{F}_{\mathrm{ACRS}}$ [12], which overcome these impossibility results, but we do not explore these advanced UC issues with respect to our OT construction in this work.

3 Preliminaries

Bilinear Groups. Let $\mathsf{BMsetup}$ be an algorithm that, on input 1^κ, outputs the parameters for a bilinear mapping as $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g \in \mathbb{G})$, where g generates $\mathbb{G}$, the groups $\mathbb{G}, \mathbb{G}_T$ each have prime order p, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. We consider the following assumptions made in these groups.

Decision Linear Assumption (DLIN) [5]: Let $\mathsf{BMsetup}(1^\kappa) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e, g) = \gamma$. For all p.p.t. adversaries Adv, the following probability is strictly less than $1/2 + 1/\mathrm{poly}(\kappa)$:
$\Pr[f, h \xleftarrow{\$} \mathbb{G}; a, b \xleftarrow{\$} \mathbb{Z}_p; z_0 \leftarrow h^{a+b}; z_1 \xleftarrow{\$} \mathbb{G}; d \leftarrow \{0,1\} : \mathsf{Adv}(\gamma, g, f, h, g^a, f^b, z_d) = d]$.

Uniform q-Hidden Strong Diffie-Hellman (q-HSDH) [8, 3]: Let $\mathsf{BMsetup}(1^\kappa) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e, g)$ $= \gamma$. For all p.p.t. adversaries Adv, the following probability is strictly less than $1/\mathrm{poly}(\kappa)$:

$$\Pr[h \xleftarrow{\$} \mathbb{G}; x, c_1, \ldots, c_q \xleftarrow{\$} \mathbb{Z}_p; (A, B, C) \leftarrow \mathsf{Adv}(\gamma, g, g^x, h, (g^{1/(x+c_1)}, g^{c_1}, h^{c_1}) \in \mathbb{G}^3, \ldots, (g^{1/(x+c_q)}, g^{c_q}, h^{c_q}) \in \mathbb{G}^3) : (A, B, C) = (g^{1/(x+c)}, g^c, h^c) \wedge c \notin \{c_1, \ldots, c_q\}].$$

Boyen and Waters did not specify the distribution for sampling the c_i values in q-HSDH [8]. Following Belenkiy *et al.* [3], we explicitly require that they be sampled uniformly from $\mathbb{Z}_p$.

q-Strong Decision Linear (q-SDLIN): Let $\mathsf{BMsetup}(1^\kappa) \rightarrow (p, \mathbb{G}, \mathbb{G}_T, e, g) = \gamma$. Let u, v, h be random elements in $\mathbb{G}$ and x_1, x_2, r_i, s_i be random values in $\mathbb{Z}_p$, then given the values $(\gamma, u, v, h, u^{x_1}, v^{x_2}, \{u^{r_i}, v^{s_i}, u^{1/(x_1+r_i)}, v^{1/(x_2+s_i)}\}_{i\in[1,q]})$, no p.p.t. adversary Adv can distinguish $\{h^{r_i+s_i}\}_{i\in[1,q]}$ from q random values in $\mathbb{G}$ with non-negligible advantage.

Note that the above assumption implies Decision Linear. The OT scheme of Camenisch *et al.* [10] is the only prior efficient, adaptive OT scheme shown to be fully-simulatable in the standard model. That construction also required both q-based decisional and computational assumptions in its proof of security. Thus, it would be very interesting to know if Decision Linear implies q-Strong Decision Linear for $q > 1$. Further, Decision Linear implies q-Strong Decision Linear for $q = 1$, but it is unknown if this implication holds for $q > 1$.

3.1 Groth-Sahai Proofs

The Groth-Sahai proof system [23] permits a variety of efficient non-interactive proofs of the satisfiability of one or more pairing product equations. For variables $\mathcal{X}_i \in \mathbb{G}$ and constants $\mathcal{A}_i \in \mathbb{G}, a_{i,j} \in \mathbb{Z}_p$, and $t_T \in \mathbb{G}_T$, these equations have the form:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \prod_{i=1}^{m} \prod_{j=1}^{m} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{i,j}} = t_T$$

Groth and Sahai show how to construct Witness Indistinguishable proof-of-knowledge of a satisfying witness to such an equation, in groups where the DLIN assumption holds. The proof system they describe can be composed over multiple equations involving the same variables. Additionally, they point out that in some special cases, their techniques can be strengthened to provide Zero Knowledge. Unlike the interactive proofs used in [10, 21], the Groth-Sahai proofs do not use adversarial rewinding in their security analysis. Therefore, we are able to use them in our protocol.

Groth-Sahai Commitments [23]. At the core of the Groth-Sahai system is a homomorphic commitment scheme to elements of $\mathbb{G}$.[2] The public parameters for the commitment scheme can be generated in one of two ways. Method (1) leads to a perfectly-binding commitment scheme, while method (2) leads to a perfectly-*hiding* scheme. Note that the two parameter distributions are computationally indistinguishable under the DLIN assumption.

When the GS commitment parameters are configured according to method (1), they are equivalent to a BBS encryption [5] of an element of $\mathbb{G}$, and can be decrypted by a party that knows a trapdoor to the commitment parameters. When commitments are configured according to method (2), a "simulation" trapdoor can be used on random commitments to open them to any value g^x for known x. We refer the reader to Appendix C for details.

The Proof System. We now describe the proof system at a high level, adopting some notation and exposition from [3]. For this description we will conceal many of the underlying details, though the reader can refer to [23, 3] for a more detailed explanation. The proof system contains the following (possibly probabilistic) polynomial time algorithms:

[2]As noted in [23, 3] commitment scheme can also be used to commit to elements of $\mathbb{Z}_p$, though we use this only in the context of simulating proofs.

$\mathsf{GSSetup}(\gamma)$. On input $\gamma \in \mathsf{BMsetup}(1^\kappa)$, outputs a string GS containing parameters for the proof system. This string embeds binding parameters for the G-S commitment scheme.

$\mathsf{GSProve}(GS, S, w)$. On input a statement S describing the equation, and a satisfying witness $W \in \langle\{\mathcal{X}\}_{1\ldots M}\rangle$, outputs a proof π. To formulate this proof, a commitment $\tilde{C}_i$ is generated for each element in W. The proof embeds openings to the commitments in such a way that a prover can ascertain that S is verifiably satisfied, and yet the elements of W remain hidden.

$\mathsf{GSVerify}(GS, \pi)$. Verifies the proof π (using the commitments and opening values) and outputs Accept if π is valid, Reject otherwise. (For compactness of notation, we will assume that π embeds the statement S).

Above we describe the proof system in normal operation. Our security proofs additionally use:

$\mathsf{GSExtractSetup}(\gamma)$. Outputs GS (distributed identically to the output of $\mathsf{GSSetup}(\gamma)$) and an extraction trapdoor td_{ext} containing a trapdoor for the commitment scheme. This trapdoor permits an extraction of a valid witness from the commitments embedded within a proof.

$\mathsf{GSExtract}(GS, td_{ext}, \pi)$. Given a proof π and the extraction trapdoor, extracts $\mathcal{X}_i$ from each commitment $\tilde{C}_i$, and outputs the witness $W = \langle\{\mathcal{X}\}_{1\ldots M}\rangle$ that satisfies the equations.

$\mathsf{GSSimulateSetup}(\gamma)$. Outputs GS' that are computationally indistinguishable from the output of $\mathsf{GSSetup}(\gamma)$, as well as a simulation trapdoor td_{sim} which consists of a simulation trapdoor for the commitment scheme.

$\mathsf{GSSimProve}(GS', td_{sim}, S)$. Given simulation parameters GS' and trapdoor td_{sim}, outputs a proof π of statement S that such that $\mathsf{GSVerify}(GS', \pi) =$ Accept. Note that this algorithm operates on certain restricted classes of statements (see below).

In the general case, Groth-Sahai proofs provide strong Witness Indistinguishability in groups where the Decision Linear assumption holds. However, in the special case where in all equations being simultaneously satisfied, the value $t_T = 1$ (or t_T can be decomposed in a specific way), then it is also possible to form proofs that meet a strong definition of composable Zero-Knowledge. We will further discuss the set of statements for which Zero-Knowledge proofs are possible below, and momentarily refer to this class as $\vec{S}_{ZK}$. We now discuss the security properties of the proof system:

Correctness. For honestly-generated GS and π, $\mathsf{GSVerify}(GS, \pi)$ will always output Accept.

Extractability (Soundness). For $(GS, td_{ext}) \in \mathsf{GSExtractSetup}(\gamma)$ and some π (embedding a statement S): if $\mathsf{GSVerify}(GS, \pi)$ outputs Accept then with probability 1 the algorithm $\mathsf{GSExtract}(GS, td, \pi)$ extracts a witness $W \in \langle\{\mathcal{X}\}_{1\ldots M}\rangle$ that satisfies S.

Composable Witness-Indistinguishability. We first require that the parameters generated by $\mathsf{GSSimulateSetup}(\gamma)$ be computationally indistinguishable from the parameters generated by $\mathsf{GSSetup}(\gamma)$. We additionally require that all *p.p.t.* adversaries $\mathcal{A}$ have advantage 0 in the following game. Hand $\mathcal{A}$ the parameters $GS' \leftarrow \mathsf{GSSimulateSetup}(\gamma)$, and allow $\mathcal{A}$ to output (S, w_0, w_1) where S is a statement and w_0, w_1 are distinct satisfying witnesses. Select $b \xleftarrow{\$} \{0, 1\}$, give $\mathcal{A}$ the proof $\pi \leftarrow \mathsf{GSProve}(GS', S, w_b)$, and collect its guess b'. $\mathcal{A}$'s advantage is defined as $|\Pr[b = b'] - 1/2|$.

Composable Zero-Knowledge. We again require that the parameters generated by $\mathsf{GSSimulateSetup}(\gamma)$ be computationally indistinguishable from the parameters generated by $\mathsf{GSSetup}(\gamma)$. We additionally require that all *p.p.t.* adversaries $\mathcal{A}$ have advantage 0 in the following game. Generate $(GS', td_{sim}) \leftarrow \mathsf{GSSimulateSetup}(\gamma)$, and give GS'

to $\mathcal{A}$. Allow $\mathcal{A}$ to output (S, w) where $S \in \vec{S}_{ZK}$ and w is a satisfying witness. Let $\pi_0 \leftarrow \mathsf{GSProve}(GS', S, w), \pi_1 \leftarrow \mathsf{GSSimProve}(GS', td_{sim}, S)$. Select $b \overset{\$}{\leftarrow} \{0,1\}$, give $\mathcal{A}$ the proof π_b, and collect its guess b'. $\mathcal{A}$'s advantage is $|\Pr[b = b'] - 1/2|$.

Note that GS proofs can be defined over multiple pairing product equations. In this case, satisfiability implies knowledge of a witness for each statement. In our constructions, we will denote a GS proof statement using the notation of Camenisch and Stadler [11]. For instance, $NIWI_{GS_S}\{(a_1, a_2) : e(a_1, a_2)/e(g, h) = 1 \ \wedge \ e(a_2, g_2)/e(d_2, a_3) = 1\}$ represents a non-interactive Witness Indistinguishable proof of knowledge, formed under parameters GS_S, of a witness $W = \langle a_1, a_2 \rangle$ that satisfies both statements. All values not in enclosed in ()'s are assumed to be known to the verifier. We will alternatively use the notation *NIZK* to denote Zero-Knowledge proofs.

Statements with Zero-Knowledge Proofs. While Groth and Sahai [23] mainly focus on Witness-Indistinguishable (WI) proofs, they note that certain classes of pairing-product statements admit Zero-Knowledge proofs as well. In order to prove a statement in Zero-Knowledge (as per the definition above), a simulator must be able to produce a simulated proof π without being given specific knowledge of a witness to the statement. Note that if the simulator can compute a valid witness by itself, then it is sufficient to simply use a WI proof. For instance, in the special case where $t_T = 1$ for a pairing product equation, the simulator can always compute a satisfying witness by selecting each $\mathcal{X}_i = g^0$.

Groth and Sahai further observe that more complex statements can be made Zero Knowledge by applying the simulation trapdoor for the Groth-Sahai commitment scheme. This trapdoor allows the simulator to open a random commitment to any g^x (for known x), and can be applied such that the same commitment is opened *differently* for each equation within the statement. In some cases, we may need to re-write a statement in order to construct a ZK proof. For example, consider the statement $\{e(a, d) = e(g, h)\}$ made on variable a and constants d, g, h. By rewriting we obtain:

$$NIZK\{(a, b) : e(a, d)e(b, h^{-1}) = 1 \ \wedge \ e(b, g)e(g^{-1}, g) = 1\}$$

Note that the statement above is equivalent to the original, by the property that $b = g$ is the only valid solution to the second equation. We can simulate the statement by opening the appropriate commitments such that $a = b = g^0$ in the first equation, while in the second equation $b = g$. We will use similar techniques to simulate the Zero-Knowledge proofs in our constructions.

4 The Construction

Our adaptive oblivious transfer protocol, $\mathsf{OT}^N_{k \times 1}$, is described in Figure 4, with each of the algorithms ($\mathsf{OTGenCRS}$, $\mathsf{OTInitialize}$, $\mathsf{OTRequest}$, $\mathsf{OTRespond}$, $\mathsf{OTComplete}$) described below.

$\mathsf{OTGenCRS}(1^\kappa)$. Given security parameter κ, generate parameters for a bilinear mapping $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMsetup}(1^\kappa)$. Compute $GS_S \leftarrow \mathsf{GSSetup}(\gamma)$ and $GS_R \leftarrow \mathsf{GSSetup}(\gamma)$. Choose random values $g_1, g_2, h \in \mathbb{G}$ and output $\mathsf{crs} = (\gamma, GS_S, GS_R, g_1, g_2, h)$.

$\mathsf{OTInitialize}(\mathsf{crs}, m_1, \ldots, m_N)$. This algorithm is executed by the Sender. On input a collection of N messages and the crs, it outputs a commitment to the database, T, for publication to the Receiver, together with a Sender secret key, sk. We treat messages as elements of $\mathbb{G}$, since there exist efficient mappings between strings in $\{0,1\}^\ell$ and elements in $\mathbb{G}$ (e.g., [6, 1]).

Protocol OTA

OTA is parameterized by the algorithms ($\mathsf{OTInitialize}$, $\mathsf{OTRequest}$, $\mathsf{OTRespond}$, $\mathsf{OTComplete}$).

When $\mathbf{S}$ is activated with $(\mathsf{sid}, \mathsf{sender}, M_1, \ldots, M_N)$:

1. $\mathbf{S}$ queries $\mathcal{F}_{CRS}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and receives $(\mathsf{sid}, \mathsf{crs})$. $\mathbf{R}$ then queries $\mathcal{F}_{CRS}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and receives $(\mathsf{sid}, \mathsf{crs})$.
2. $\mathbf{S}$ obtains $(T, sk) \leftarrow \mathsf{OTInitialize}(\mathsf{crs}, M_1, \ldots, M_N)$, sends $(\mathsf{sid}, \mathsf{ssid}, T)$ to $\mathbf{R}$ and stores $(\mathsf{sid}, \mathsf{ssid}, T, sk)$.

When $\mathbf{R}$ is activated with $(\mathsf{sid}, \mathsf{receiver}, \sigma)$, and $\mathbf{R}$ has previously received $(\mathsf{sid}, \mathsf{ssid}, T)$ and $(\mathsf{sid}, \mathsf{crs})$:

1. $\mathbf{R}$ runs $(Q, Q_{priv}) \leftarrow \mathsf{OTRequest}(\mathsf{crs}, T, \sigma)$, sends $(\mathsf{sid}, \mathsf{ssid}, Q)$ to $\mathbf{S}$ and stores $(\mathsf{sid}, \mathsf{ssid}, Q_{priv})$.
2. $\mathbf{S}$ gets $(\mathsf{sid}, \mathsf{ssid}, Q)$ from $\mathbf{R}$, runs $R \leftarrow \mathsf{OTRespond}(\mathsf{crs}, T, sk, Q)$, and sends $(\mathsf{sid}, \mathsf{ssid}, R)$ to $\mathbf{R}$.
3. $\mathbf{R}$ receives $(\mathsf{sid}, \mathsf{ssid}, R)$ from $\mathbf{S}$, and outputs $(\mathsf{sid}, \mathsf{ssid}, \mathsf{OTComplete}(\mathsf{crs}, T, R, Q_{priv}))$.

Figure 4: A high-level outline of the $\mathsf{OT}^N_{k\times 1}$ protocol, with details of each algorithm described in Section 4. We make no explicit mention of the value k, the total transfers permitted by the Sender, because our protocol does not depend on it. The Sender may choose to stop answering the Receiver's queries at any point, in which case $\mathsf{OTRespond}$ outputs "reject" and $\mathsf{OTComplete}$ accepts this as the message $\perp$.

1. Choose random values $x_1, x_2, \alpha_1, \alpha_2, \alpha_3 \in \mathbb{Z}_p$.
2. Set $(u_1, u_2) \leftarrow (h^{1/x_1}, h^{1/x_2})$, and $pk \leftarrow (u_1, u_2, u_1^{\alpha_1}, u_2^{\alpha_2}, g_2^{\alpha_3})$.
3. For $j = 1, \ldots, N$ encrypt each message m_j as:
   (a) Select random $r, s, t \in \mathbb{Z}_p$.
   (b) Set $C_j \leftarrow \Big(u_1^r,\ u_2^s,\ g_1^r,\ g_2^s,\ m_j \cdot h^{r+s},\ u_1^{1/(\alpha_1+r)},\ u_2^{1/(\alpha_2+s)},\ g_2^t,\ (u_1^r u_2^s h)^t g_1^{\alpha_3}\Big)$.
4. Set $T \leftarrow (pk, C_1, \ldots, C_N)$ and $sk \leftarrow (x_1, x_2)$. Output (T, sk).

Notice that the value T has a structure that can be publicly verified. Represent pk as $(p_1, \ldots, p_5)$. Parse each ciphertext C_i as $(c_1, \ldots, c_9)$ and check that the following conditions hold:

$$\begin{aligned} e(p_1, c_3) = e(c_1, g_1) \quad &, \quad e(p_2, c_4) = e(c_2, g_2) \\ e(c_6, p_3 \cdot c_1) = e(p_1, p_1) \quad &, \quad e(c_7, p_4 \cdot c_2) = e(p_2, p_2) \\ &, \quad e(g_2, c_9)/e(c_8, c_1 \cdot c_2 \cdot h) = e(g_1, p_5). \end{aligned}$$

$\mathsf{OTRequest}(\mathsf{crs}, T, \sigma)$. This algorithm is executed by a Receiver. On input T generated by the Sender, along with an item index σ, generates a query Q for transmission to the Sender.

1. Parse T as $(pk, C_1, \ldots, C_N)$, and ensure that it is correctly formed (see above). If T is not correctly formed, abort the protocol. This check need be done only once.
2. Parse crs as $(\gamma, GS_S, GS_R, g_1, g_2, h)$, pk as $(p_1, \ldots, p_5)$, and C_σ as $(c_1, \ldots, c_9)$.
3. Select random $v_1, v_2 \in \mathbb{Z}_p$ and set $d_1 \leftarrow (c_1 \cdot p_1^{v_1}), d_2 \leftarrow (c_2 \cdot p_2^{v_2})$, $t_1 \leftarrow h^{v_1}$, $t_2 \leftarrow h^{v_2}$.

4. Use the Groth-Sahai techniques and reference string GS_R to compute the Witness-Indistinguishable proof of values pertaining to the ciphertext C_σ (which the Receiver wishes to have the Sender help him open) and blinding values:

$$\begin{aligned}\pi = NIWI_{GS_R}\{(c_1, c_2, c_3, c_4, c_6, c_7, c_8, c_9, t_1, t_2) :&\\ e(c_6, p_3 \cdot c_1) = e(p_1, p_1) \ \wedge\ e(p_1, c_3)e(c_1, g_1^{-1}) = 1 \ \wedge&\\ e(c_7, p_4 \cdot c_2) = e(p_2, p_2) \ \wedge\ e(p_2, c_4)e(c_2, g_2^{-1}) = 1 \ \wedge&\\ e(d_1 \cdot c_1^{-1}, h)e(p_1^{-1}, t_1) = 1 \ \wedge\ e(d_2 \cdot c_2^{-1}, h)e(p_2^{-1}, t_2) = 1 \ \wedge&\\ e(g_2, c_9)e(c_8, c_1 \cdot c_2 \cdot h)^{-1} = e(g_1, p_5)\}&\end{aligned}$$

To explain what is happening in this statement, first observe that the second and fourth equations ensure that (p_1, g_1, c_1, c_3) and (p_2, g_2, c_2, c_4) are both DDH tuples. Thus, for some values of $r, s \in \mathbb{Z}_p$, we have that $p_1^r = c_1$, $g_1^r = c_3$, $p_2^s = c_2$ and $g_2^s = c_4$. Under this characterization of (c_1, c_2) and with $(p_1, \ldots, p_5)$ all public, the first and third equations ensure that $c_6 = p_1^{1/(\alpha_1+r)}$ and $c_7 = p_2^{1/(\alpha_2+s)}$, where $p_3 = p_1^{\alpha_1}$ and $p_4 = p_2^{\alpha_2}$ for some values $\alpha_1, \alpha_2 \in \mathbb{Z}_p$. The next two equations guarantee that if we view $d_1 = p_1^{v_1+r}$ and $d_2 = p_2^{v_2+s}$, for some values $v_1, v_2 \in \mathbb{Z}_p$, then $t_1 = h^{v_1}$ and $t_2 = h^{v_2}$. Finally, the last equation ensures that if we represent $c_8 = g_2^t$ and $p_5 = g_2^{\alpha_3}$ for some t, α_3, then $c_9 = (c_1c_2h)^t \cdot g_1^{\alpha_3}$. These checks guarantee that the witness used by the Receiver, and thus the decryption request being made, corresponds to one of the N ciphertexts published by the Sender.

5. Set request $Q \leftarrow (d_1, d_2, \pi)$, and private state $Q_{priv} \leftarrow (Q, \sigma, v_1, v_2)$. Output (Q, Q_{priv}).

$\mathsf{OTRespond}(\mathsf{crs}, T, sk, Q)$. This algorithm is executed by the Sender. If the Sender does not wish to answer any more requests for the Receiver, then the Sender outputs the message "reject". Otherwise, the Sender processes the Receiver's request Q as:

1. Parse crs as $(\gamma, GS_S, GS_R, g_1, g_2, h)$, T as $(pk, C_1, \ldots, C_N)$, and sk as (x_1, x_2).
2. Parse pk (from T) as $(p_1, \ldots, p_5)$.
3. Parse Q as (d_1, d_2, π) and verify proof π using GS_R. Abort if verification fails.
4. Set $a_1 \leftarrow d_1^{x_1}$, $a_2 \leftarrow d_2^{x_2}$, and $s \leftarrow a_1 \cdot a_2$.
5. Use the Groth-Sahai techniques and reference string GS_S to formulate a zero-knowledge proof that the decryption value s is properly computed:

$$\begin{aligned}\delta = NIZK_{GS_S}\{(a_1, a_2, a_3) : e(a_1, p_1)e(d_1^{-1}, a_3) = 1 \ \wedge\ e(a_2, p_2)e(d_2^{-1}, a_3) = 1 \ \wedge&\\ e(s, a_3)e(a_1 \cdot a_2, h^{-1}) = 1 \ \wedge\ e(g, a_3) = e(g, h)\}&\end{aligned}$$

Observe that the last equation ensures that $a_3 = h$. The third equation ensures that $s = a_1 \cdot a_2$, while the first two, since the values (p_1, d_1, p_2, d_2, h) are known to both parties, ensure that $a_1 = d_1^{x_1}$ and $a_2 = d_2^{x_2}$. This guarantees that s is correctly formed.

6. Output $R \leftarrow (s, \delta)$.

$\mathsf{OTComplete}(\mathsf{crs}, T, R, Q_{priv})$. This algorithm is executed by the Receiver. On input R generated by the Sender in response to a request Q, along with state Q_{priv}, outputs a message m or $\perp$. If R is the message "reject", then the Receiver outputs $\perp$. Otherwise, the Receiver does:

1. Parse crs as $(\gamma, GS_S, GS_R, g_1, g_2, h)$, T as $(pk, C_1, \ldots, C_N)$, R as (s, δ), and Q_{priv} as (Q, σ, v_1, v_2).
2. Verify proof δ using GS_S. If verification fails, output $\perp$.
3. Parse C_σ as $(c_1, c_2, c_3, c_4, c_5, \ldots)$ and output $m = c_5/(s \cdot h^{-v_1} \cdot h^{-v_2})$.

4.1 Efficiency Analysis

When the protocol in Figure 4 is implemented using the algorithms described above, we obtain a $(k + 1/2)$-round protocol with communications cost $O(N + k)$, where $k \leq N$. More concretely, the crs is comprised of 15 elements in $\mathbb{G}$, the Sender's public key contains 5 elements in $\mathbb{G}$, and each of the N ciphertexts in T requires 9 elements in $\mathbb{G}$. Moreover, each item transfer involves transmission of 95 elements of $\mathbb{G}$ from Receiver to Sender, and then 46 elements of $\mathbb{G}$ from Sender to Receiver.

The message space of our OT protocol is elements in $\mathbb{G}$, which will be sufficient for transferring a symmetric encryption key to unlock a file of arbitrary size.

4.2 Security Analysis

Theorem 4.1 *Instantiated with the above algorithms,* OTA *securely realizes the functionality* $\mathcal{F}_{OT}^{N\times 1}$ *in the* $\mathcal{F}_{CRS}$-hybrid *model under the q-Strong Decision Linear and uniform q-HSDH assumptions.*

Let us now provide some intuition behind this proof, with the details contained in Appendix A. When either the Sender or the Receiver is corrupted, we wish to describe a simulator $\mathcal{S}$ such that it can interact with the ideal functionality $\mathcal{F}_{OT}^{N\times 1}$ (which we'll denote simply as $\mathcal{F}$) and the environment $\mathcal{Z}$ appropriately; i.e., $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA},\mathcal{A},\mathcal{Z}}$.

We begin with the case where the real world adversary $\mathcal{A}$ corrupts the Sender, and thus $\mathcal{S}$ must interact with $\mathcal{F}$ as the ideal Sender and with (an internal copy of) $\mathcal{A}$ as a real-world Receiver. Here $\mathcal{S}$ does the following:

1. Ask $\mathcal{A}$ to begin an OT protocol, and set the crs *for these two parties* by running $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $GS_S \leftarrow \mathsf{GSSetup}(\gamma)$, $GS_R \leftarrow \mathsf{GSSetup}(\gamma)$, and selecting random elements $h \in \mathbb{G}$ and $a_1, a_2 \in \mathbb{Z}_p$. Set $\mathsf{crs} = (\gamma, GS_S, GS_R, h^{a_1}, h^{a_2}, h)$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.
2. Obtain the database commitment T from $\mathcal{A}$. Verify that T is well-formed, abort if not. Otherwise, use a_1, a_2 to decrypt each ciphertext $C_i = (c_1, \ldots, c_9)$ as $m_i = c_5/(c_3^{1/a_1} \cdot c_4^{1/a_2})$. Map each element $m_i \in \mathbb{G}$ to a string in $\{0,1\}^\ell$ [1]. Send $(\mathsf{sid}, \mathbf{S}, m_1, \ldots, m_N)$ to $\mathcal{F}$.
3. Upon receiving $(\mathsf{sid}, \mathsf{request})$ from $\mathcal{F}$, choose a random index $\sigma \in [1, N]$ and return $\mathsf{OTRequest}(\mathsf{crs}, T, \sigma)$ to $\mathcal{A}$. This response includes two random values d_1, d_2 and a non-interactive witness indistinguishable proof with respect to $GS_R \in \mathsf{crs}$ that d_1, d_2 correspond to a ciphertext C_σ. This proof can be performed honestly and without rewinding.
4. If $\mathcal{A}$ issues a "reject" message or responds with anything other than a value in $\mathbb{G}$ and a valid NIZK proof, then $\mathcal{S}$ tells $\mathcal{F}$ to fail the request by sending message $(\mathsf{sid}, 0)$. Otherwise, $\mathcal{S}$ sends the message $(\mathsf{sid}, 1)$ to $\mathcal{F}$.

The indistinguishability argument here follows from the indistinguishability of the crs (from a real crs), the perfect extraction of the messages m_i, and the WI proof during each request phase, which guarantees that $\mathcal{A}$ (the corrupt Sender) cannot be selectively choosing to fail based on the Receiver's choices. Thus, $\mathcal{S}$ can adequately mimic its response pattern.

Next, we consider the case where the real world adversary $\mathcal{A}$ corrupts the Receiver, and thus $\mathcal{S}$ must interact with $\mathcal{F}$ as the ideal Receiver and with (and internal copy of) $\mathcal{A}$ as real-world Receiver. This case requires that the $q = N$ for the uniform q-HSDH assumption. Here $\mathcal{S}$ does the following:

1. Ask $\mathcal{A}$ to begin an OT protocol, and set the crs *for these two parties* by running $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $(GS_S, td_{sim}) \leftarrow \mathsf{GSSimulateSetup}(\gamma)$ and $(GS_R, td_{ext}) \leftarrow \mathsf{GSExtractSetup}(\gamma)$. Select random $g_1, g_2, h \in \mathbb{G}$. Set $\mathsf{crs} \leftarrow (\gamma, GS_S, GS_R, g_1, g_2, h)$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.
2. $\mathcal{S}$ must commit to a database of messages for $\mathcal{A}$ *without* knowing the messages $m_1, \ldots, m_N$. Thus, $\mathcal{S}$ simply commits to arbitrary junk messages, and sends the corresponding T to $\mathcal{A}$.
3. When $\mathcal{A}$ makes a transfer request, $\mathcal{S}$ uses td_{ext} to extract the witness corresponding to the index σ from the NIWI proof. (This extraction is done via opening perfectly-binding commitments which are includes in the WI proof and does not require any rewinding.)
4. $\mathcal{S}$ now sends $(\mathsf{sid}, \mathbf{R}, \sigma)$ to $\mathcal{F}$ to obtain the real m_σ message.
5. Now, $\mathcal{S}$ returns a response to $\mathcal{A}$ which opens C_σ to m_σ and then uses td_{sim} to simulate an NIZK proof that this opening is correct. The NIZK proof here is designed in such a way that simulation is always possible and no rewinding is necessary.

The indistinguishability argument here follows from the indistinguishability of the crs (from a real crs), the indistinguishability of the "fake" database T, the ability to extract witnesses from the NIWI proofs, and the zero-knowledge property of "fake" NIZK proofs. Notice that $\mathcal{S}$ is never *both* simulating and extracting via the same (subsection of the) common reference string; indeed, we do not require that the proofs be simulation-sound.

Sampling from a Common Random String. We briefly note that by the same arguments used above, the Reference String used in our construction can be replaced with a Common Random String. Note that crs embeds $(\gamma, GS_S, GS_R, g_1, g_2, h)$, for $GS_R, GS_S \in \mathsf{GSSetup}(\gamma)$ and $g_1, g_2, h \in_R \mathbb{G}$. Per Appendix C, each set of Groth-Sahai commitment parameters embeds a tuple of the form $(g, g^a, g^b, g^{as}, g^{bz}, g^w) \in \mathbb{G}^6$, for $a, b, s, z \in_R \mathbb{Z}_p$. When $\mathsf{GSSetup}$ is used, the parameters are generated such that $w = s + z$. When $\mathsf{GSSimulateSetup}$ is used, w is uniformly distributed in $\mathbb{Z}_p$. Under DLIN, the latter distribution is indistinguishable from the correct one, and thus we may sample the components of GS_S, GS_R uniformly from $\mathbb{G}$. Since the parameters γ can be sampled from a random string [22], then all elements of crs can therefore be derived from a uniformly *random* string when a source of common randomness is available.

5 On Multiple Receivers

OT is traditionally described as a two-party protocol between a Sender and Receiver. We presented our main construction in this setting. However, since we are motivated by the application of OT to database systems, we would also like to support applications where multiple users share a single database. Naively this can be accomplished by requiring the database to run separate OT protocol instances with each user. However, this approach can be quite inefficient, and moreover does not ensure *consistency* in the database viewed by individual Receivers. Consider a strengthening of the security definition of $\mathcal{F}_{OT}^{N \times 1}$ (in Figure 3) to include the additional requirement that all Receivers "view" the same database, i.e., the database owner cannot selectively alter the messages in the database when interacting with different receivers – on query σ from *any* receiver, he must return a value in $\{m_\sigma, \perp\}$. Fortunately, *consistency* is easy and inexpensive to achieve in our construction

– simply alter $\mathcal{F}_{CRS}^{\mathcal{D},P}$ to return the *same* values (g_1, g_2, h) as part of the crs to *all* receivers and have the Sender publish one database commitment T to everyone, handling joint state via [17]. Intuitively, this captures consistency because the simulator can set the values (g_1, g_2, h) and then trapdoor decrypt all messages in T (see Appendix B). Given the soundness of the GS proofs, all of the Sender's responses to any Receiver must be consistent with T, even if the other parts of their common reference strings are distinct. Note that it is not at all clear how *consistency* can be achieved efficiently *even in the non-adaptive setting* using prior UC results [28], since there each Receiver provides her own encryption key for the Sender to bundle the messages in.

References

[1] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In *CCS '05*, pages 92–101. ACM Press, 2005.

[2] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage from keyword searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005.

[3] Mira Belenkiy, Melissa Chase, Markulf Kolweiss, and Anna Lysyanskaya. Non-interactive anonymous credentials. To appear in TCC '08. Available from the IACR Cryptology ePrint Archive as Report 2007/384, 2008.

[4] Dan Boneh and Xavier Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 223–238, 2004.

[5] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO '04*, volume 3152 of LNCS, pages 45–55, 2004.

[6] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *CRYPTO '01*, volume 2139 of LNCS, pages 213–229, 2001.

[7] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC '05*, volume 3378 of LNCS, pages 325—341. Springer, 2005.

[8] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC '07*, volume 4450 of LNCS, pages 1–15. Springer, 2007.

[9] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *CRYPTO '86*, volume 263 of LNCS, pages 234–238, 1986.

[10] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT '07*, volume 4515 of LNCS, pages 573–590, 2007.

[11] Jan Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, volume 1296 of LNCS, pages 410–424, 1997.

[12] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with pre-existing setup. In *TCC '07*, volume 4392 of LNCS, pages 61–85, 2007.

[13] Ran Canetti. Universally Composable Security: A new paradigm for cryptographic protocols. In *FOCS '01*, page 136. IEEE Computer Society, 2001. `http://eprint.iacr.org/2000/067`.

[14] Ran Canetti. Universally composable security: Towards the bare bones of trust. In *Asiacrypt '07*, volume 4833 of LNCS, pages 88–112, 2007.

[15] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO '01*, volume 2139 of LNCS, pages 19–40. Springer, 2001.

[16] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC '02*, pages 494–503. ACM Press, 2002.

[17] Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO '03*, volume 2729 of LNCS, pages 265–281. Springer, 2003.

[18] Cheng-Kang Chu and Wen-Guey Tzeng. Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries. In *PKC '05*, volume 3386 of LNCS, pages 172–183, 2005.

[19] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *CRYPTO '82*, pages 205–210, 1982.

[20] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC '87*, pages 218–229, 1987.

[21] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT '07*, volume 4833 of LNCS, pages 265–282, 2007.

[22] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT '06*, volume 4004 of LNCS, pages 339–358. Springer, 2006.

[23] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Manuscript, 2007. Available at `http://www.brics.dk/~jg/WImoduleFull.pdf`.

[24] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC '88*, pages 20–31, 1988.

[25] Yehuda Lindell. Efficient fully-simulatable oblivious transfer. In *CT-RSA '08 (to appear)*, 2008. Available at `http://eprint.iacr.org/2008/035`.

[26] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO '99*, volume 1666 of LNCS, pages 573–590, 1999.

[27] Wakaha Ogata and Kaoru Kurosawa. Oblivious keyword search. *Special issue on coding and cryptography Special issue on coding and cryptography Journal of Complexity*, 20(2-3):356–371, 2004.

[28] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. Cryptology ePrint Archive, Report 2007/348, 2007. `http://eprint.iacr.org/2007/348.pdf`.

[29] Michael Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[30] Mike Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number, 2002. Available at `http://eprint.iacr.org/2002/164`.

[31] Andrew Yao. How to generate and exchange secrets. In *FOCS '86*, pages 162–167, 1986.

A Proof of Theorem 4.1

Notation: Let $\nu(\cdot)$ denote a *negligible* function where for every polynomial $p(\cdot)$, there exists an N such that for all integers $n > N$, it holds that $\nu(n) < 1/p(n)$.

Proof. Let $\mathcal{A}$ be a static adversary that interacts with parties $\mathbf{S}, \mathbf{R}$ running protocol OTA. We construct an adversary $\mathcal{S}$ for the ideal functionality $\mathcal{F}_{OT}^{N\times 1}$. $\mathcal{S}$ begins by invoking a copy of $\mathcal{A}$ and running a simulated interaction with the environment $\mathcal{Z}$ and the parties running the protocol. $\mathcal{S}$ proceeds as follows.

Simulating the communication with $\mathcal{Z}$. Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written into the adversary $\mathcal{A}$'s input tape. Similarly, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape (to be read by $\mathcal{S}$'s environment $\mathcal{Z}$).

Simulating the case where only R is corrupted. Let $\gamma \leftarrow \mathsf{BMsetup}(1^\kappa)$, then compute $(GS'_S, td_{sim}) \leftarrow \mathsf{GSSimulateSetup}(\gamma)$ and $(GS'_R, td_{ext}) \leftarrow \mathsf{GSExtractSetup}(\gamma)$. Select $g_1, g_2, h \in_R \mathbb{G}$. Set $\mathsf{crs} \leftarrow (\gamma, GS'_S, GS'_R, g_1, g_2, h)$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.

$\mathcal{S}$ initiates the communication with $\mathcal{A}$ by generating a random message database $\hat{m}_1, \ldots, \hat{m}_N \xleftarrow{\$} \mathbb{G}$, computing $T \leftarrow \mathsf{OTInitialize}(\mathsf{crs}, \hat{m}_1, \ldots, \hat{m}_N)$ and sending (sid, T) to $\mathcal{A}$ as if from $\mathbf{S}$. Next, whenever $\mathcal{A}$ outputs (sid, Q), $\mathcal{S}$ performs as follows. First, it parses Q as (d_1, d_2, π) and (if π is valid) computes $\mathsf{GSExtract}(\mathsf{crs}, td_{ext}, \pi)$ to extract a satisfying witness $(\omega_1, \ldots, \omega_{10})$. Parse T as $(pk, C_1, \ldots, C_N)$, and for each ciphertext $C_i = (c_1, \ldots, c_9)$ determine whether $(\omega_1, \omega_2) = (c_1, c_2)$. If no matching ciphertext is found (or multiple ciphertexts match), then $\mathcal{S}$ aborts the simulation and gives no further messages to $\mathcal{A}$.

Otherwise, let σ' be the index of the matching ciphertext: $\mathcal{S}$ sends $(\mathsf{sid}, \mathsf{receiver}, \sigma')$ to $\mathcal{F}_{OT}^{N\times 1}$. When $\mathcal{F}_{OT}^{N\times 1}$ outputs $(\mathsf{sid}, m_{\sigma'})$ for $m_{\sigma'} \neq \perp$, $\mathcal{S}$ formulates the response $s = (c_5\omega_9\omega_{10})/m_{\sigma'}$ and—using the simulation trapdoor td_{sim}— simulates the zero-knowledge proof δ' indicating that s is correctly formed according to the statement defined in the $\mathsf{OTRespond}$ algorithm (see Lemma A.7 for details on simulating this proof). $\mathcal{S}$ then sends $R \leftarrow (s, \delta')$ to $\mathcal{A}$ as if from $\mathbf{S}$. $\mathcal{S}$ repeats this process for each request received from $\mathcal{A}$.

Simulating the case where only S is corrupted. Our simulation proceeds as follows. Let $\gamma \leftarrow \mathsf{BMsetup}(1^\kappa)$, then compute $GS_S \leftarrow \mathsf{GSSetup}(\gamma)$ and $GS_R \leftarrow \mathsf{GSSetup}(\gamma)$. Select $g_1, g_2, h \in_R \mathbb{G}$ such that $g_1^{y_1} = g_2^{y_2} = h$ for (y_1, y_2) known to the simulator. Set $\mathsf{crs} \leftarrow (\gamma, GS_S, GS_R, g_1, g_2, h)$. When the parties query $\mathcal{F}_{CRS}$, return $(\mathsf{sid}, \mathsf{crs})$.

$\mathcal{S}$ activates $\mathcal{A}$ and receives the message (sid, T) that would be $\mathcal{A}$'s first move in a real execution with $\mathbf{R}$. $\mathcal{S}$ verifies that T is correctly-structured, using the public check described in §4. (If T does not pass this check, $\mathcal{S}$ will instruct $\mathcal{F}_{OT}^{N\times 1}$ to fail on all message requests from $\mathbf{R}$.) Otherwise, $\mathcal{S}$ parses T as $(pk, C_1, \dots, C_N)$ and for $i = 1$ to N first parses ciphertext C_i into $(c_1, \dots, c_9)$, then computes $m'_i \leftarrow c_5/(c_3^{y_1}c_4^{y_2})$. $\mathcal{S}$ decodes each $m'_1, \dots, m'_N$ to a value in $\{0,1\}^\ell$ and sends $(\mathsf{sid}, \mathsf{sender}, m'_1, \dots, m'_N)$ to $\mathcal{F}_{OT}^{N\times 1}$.

Whenever $\mathcal{F}_{OT}^{N\times 1}$ outputs (sid) to the dummy $\mathbf{S}$ (indicating that $\mathbf{R}$ has initiated a transfer request), $\mathcal{S}$ computes $(Q, Q_{priv}) \leftarrow \mathsf{OTRequest}(\mathsf{crs}, T, 1)$ and hands (sid, Q) to $\mathcal{A}$ as if from $\mathbf{R}$. When $\mathbf{S}$ returns (sid, R), $\mathcal{S}$ checks whether $\mathsf{OTComplete}(\mathsf{crs}, T, R, Q_{priv}) = \perp$. If so, then $\mathcal{S}$ sets $b \leftarrow 0$, and $b \leftarrow 1$ otherwise. $\mathcal{S}$ returns (sid, b) to $\mathcal{F}_{OT}^{N\times 1}$.

Simulating the case where neither party is corrupted. When $\mathcal{S}$ receives k messages of the form (sid, b_i) indicating that transfers have occurred, $\mathcal{S}$ generates a simulated transcript between the honest $\mathbf{S}$ and $\mathbf{R}$. In this case, $\mathcal{S}$ runs the protocol as specified, using as $\mathbf{S}$'s input the random database $(\hat{m}_1, \dots, \hat{m}_N)$, and (for each transfer), $\mathbf{R}$'s input $\sigma = 1$. If in the i^{th} transfer $b_i = 0$ then $\mathbf{S}$'s responds with an invalid R (the empty string). Else, $\mathbf{S}$ returns a valid response as in the protocol.

Simulating the case where both parties are corrupted. In this case $\mathcal{S}$ knows the inputs to $\mathbf{S}$ and $\mathbf{R}$ and can simulate a protocol execution by generating the real messages exchanged between the two parties.

We now address the environment's ability to distinguish the ideal execution from the real protocol execution. This is shown via the following claims.

Claim A.1 *When $\mathcal{A}$ corrupts only $\mathbf{R}$, then $\mathsf{IDEAL}_{\mathcal{F}_{OT}^{N\times 1}, \mathcal{S}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA}, \mathcal{A}, \mathcal{Z}}$ under the N-*Strong DLIN *and N-*HSDH *assumptions.*

Proof. Consider the simulation described above. We will begin with the real-world protocol execution, where $\mathbf{R}$ interacts with an honest $\mathbf{S}$ that knows the message database. We will then show via a series of hybrids that the real execution transcript is computationally indistinguishable from the simulated transcript. For notational convenience, we define $\mathbf{Pr}\,[\,\mathbf{Game\ i}\,]$ as the probability that environment $\mathcal{Z}$ distinguishes the transcript of **Game** i from that of the real execution. We now describe the cases:

Game 0. This is the real-world protocol execution, where **R** interacts with an honest **S** running protocol OTA on message database $(m_1, \ldots, m_N)$. Clearly $\mathbf{Pr}[\,\mathbf{Game\ 0}\,] = 0$.

Game 1 (Parameter switching). This execution proceeds as above, except that we compute $(GS'_S, td_{sim}) \in \mathsf{GSSimulateSetup}(\gamma)$, $(GS'_R, td_{ext}) \in \mathsf{GSExtractSetup}(\gamma)$, and substitute GS'_S, GS'_R in place of the honestly-generated parameters GS_S, GS_R (td_{sim}, td_{ext} are not revealed). When the parties query $\mathcal{F}_{CRS}$, return $\mathsf{crs} = (\gamma, GS'_S, GS'_R, g_1, g_2, h)$. Note that if the DLIN assumption holds in $\mathbb{G}$, then $(GS'_S, GS'_R) \overset{c}{\approx} (GS_S, GS_R)$ (Lemma A.5) and thus $|\mathbf{Pr}[\,\mathbf{Game\ 1}\,] - \mathbf{Pr}[\,\mathbf{Game\ 0}\,]| \leq \nu(\kappa)$.

Game 2 (Extracting R's selections). This execution proceeds as above, except that for transfer phase $i = 1$ to k, we compute a candidate for **R**'s selection σ'_i by extracting from its PoK π. Parse **R**'s i^{th} request (sid, Q_i) to obtain (d_1, d_2, π) and (provided that π is valid) run $\mathsf{GSExtract}(\mathsf{crs}, td_{ext}, \pi)$ to extract a satisfying witness $(\omega_1, \ldots, \omega_{10})$. Parse T as $(pk, C_1, \ldots, C_N)$, and for each ciphertext $C_i = (c_1, \ldots, c_9)$ determine whether $(\omega_1, \omega_2) = (c_1, c_2)$. Let σ'_i be the index of the matching ciphertext. If no matching ciphertext is found (or multiple ciphertexts match), then output Extract-Fail to $\mathcal{Z}$ and send no further messages to **R**. By Lemma A.6, this event will occur with negligible probability under the DLIN and N-HSDH assumptions; therefore $|\mathbf{Pr}[\,\mathbf{Game\ 2}\,] - \mathbf{Pr}[\,\mathbf{Game\ 1}\,]| \leq \nu(\kappa)$.

Game 3 (Simulating S's responses). This execution proceeds as above, except that we will formulate each of **S**'s transfer responses independently of sk. In the previous hybrid we extracted from each of **R**'s requests to obtain a witness $(\omega_1, \ldots, \omega_{10})$ and identified a candidate σ'_i for **R**'s selection. We parse $C_{\sigma'_i}$ as $(c_1, \ldots, c_9)$ and compute $s' = (c_5 \omega_9 \omega_{10})/m_{\sigma'_i}$. Let S be the statement proved by the Sender during the $\mathsf{OTRespond}$ algorithm: we compute a simulated PoK $\delta' \leftarrow \mathsf{GSSimProve}(GS_S, td_{sim}, S)$ and set $R' \leftarrow (s', \delta')$. Note that in order to simulate a response during transfer i, it is only necessary to know the subset of messages, $(m_{\sigma'_1}, \ldots, m_{\sigma'_i})$. By Lemma A.7, the transcript including these responses is computationally indistinguishable from the distribution with valid PoKs. Thus $|\mathbf{Pr}[\,\mathbf{Game\ 3}\,] - \mathbf{Pr}[\,\mathbf{Game\ 2}\,]| \leq \nu(\kappa)$.

Game 4 (Substituting the ciphertexts). This execution proceeds as above, except that we replace **S**'s first message with (sid, T') where $T' \in \mathsf{OTInitialize}(\mathsf{crs}, \hat{m}_1, \ldots, \hat{m}_N)$ for $\hat{m}_1, \ldots, \hat{m}_N \overset{\$}{\leftarrow} \mathbb{G}$. For $i = 1$ to k, we also modify the i^{th} transfer phase such that **S**'s response is (sid, R'_i) for $R'_i = (s', \delta')$ computed as in **Game 3**, except that we must now compute the PoK δ on a possibly invalid statement S. By Lemma A.8, the hardness of N-Strong DLIN implies that the distribution of messages is indistinguishable from the real execution, even though s' may be *incorrectly* formed with respect to S. Thus $|\mathbf{Pr}[\,\mathbf{Game\ 4}\,] - \mathbf{Pr}[\,\mathbf{Game\ 3}\,]| \leq \nu(\kappa)$.

Notice that the distribution produced in **Game 4** is identical to that of our simulation. By summation, we have that $\mathbf{Pr}[\,\mathbf{Game\ 4}\,] \leq \nu(\kappa)$ and thus $\mathsf{IDEAL}_{\mathcal{F}_{OT}^{N\times 1}, \mathcal{S}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA}, \mathcal{A}, \mathcal{Z}}$ under the N-HSDH and N-Strong DLIN assumptions.

□

Claim A.2 *When $\mathcal{A}$ corrupts only* **S***, then $\mathsf{IDEAL}_{\mathcal{F}_{OT}^{N\times 1}, \mathcal{S}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA}, \mathcal{A}, \mathcal{Z}}$ under the DLIN assumption.*

Proof. Consider the simulation described above. Again we begin with the real-world protocol execution, where **S** interacts with an honest **R** that chooses messages according to an arbitrary selection strategy Σ. We then show via a series of hybrids that the real execution transcript is computationally indistinguishable from the simulated transcript.

> **Game 0.** This is the real-world protocol execution, where **S** interacts with an honest **R** running protocol OTA using selection strategy Σ. Clearly $\mathbf{Pr}[\,\mathbf{Game\ 0}\,] = 0$.
>
> **Game 1 (Parameter generation).** This execution proceeds as above, except that we select g_1, g_2, h such that $g_1^x = g_2^y = h$ for known (x, y). When the parties query $\mathcal{F}_{CRS}$, return $\mathsf{crs} = (\gamma, GS_S, GS_R, g_1, g_2, h)$. Note that the distribution of g_1, g_2, h is identical to the normal distribution. Thus $|\mathbf{Pr}[\,\mathbf{Game\ 1}\,] - \mathbf{Pr}[\,\mathbf{Game\ 0}\,]| = 0$.
>
> **Game 2 (Substituting R's queries).** Next, during transfer $i = 1$ to k, we modify the transcript by generating $Q_i' \leftarrow \mathsf{OTRequest}(T, 1)$ and replacing **R**'s request with (sid, Q_i'). Let $Q' = (d_1', d_2', \pi')$. Observe that for any $i \in [1, N]$, where $C_i = (c_1, \ldots, c_9)$, we can express d_1', d_2' as $c_1 p_1^{v_1'}, c_2 p_2^{v_2'}$ for some v_1', v_2'. Thus for every C_i there exists a witness $(c_1, c_2, c_3, c_4, c_6, c_7, c_8, c_9, h^{v_1'}, h^{v_2'})$ that satisfies the pairing product equation S_π. By the Witness-Indistinguishability property of the Groth-Sahai proof system, the value Q_1' is indistinguishable from a request formed on a different $\sigma_j \in [1, N]$. Thus $|\mathbf{Pr}[\,\mathbf{Game\ 2}\,] - \mathbf{Pr}[\,\mathbf{Game\ 1}\,]| \le \nu(\kappa)$.

Game 2 has an identical distribution to our simulation, and by summation $\mathbf{Pr}[\,\mathbf{Game\ 2}\,] \le \nu(\kappa)$. It remains to show that the in our simulation the distribution of messages obtained by an ideal **R** interacting with $\mathcal{F}_{OT}^{N\times 1}$ are identical to the messages recovered by an honest **R** running the protocol directly with **S**. This implies that for every set of indices $(\sigma_1, \ldots, \sigma_k)$ the plaintexts $(m_{\sigma_1}', \ldots, m_{\sigma_k}')$ obtained by $\mathcal{S}$— which decrypts the ciphertexts in T with the trapdoor (x, y)— are identical to the messages recovered by an honest **R** running the protocol with **S**.

S's initial output T embeds $pk = (p_1, \ldots, p_5)$. Let (a, b) be **S**'s secret key, which is implicitly defined by $p_1^a = p_2^b = h$. We observe that if T passes the validity check run by honest **R**, then each ciphertext C_i can be expressed as $(p_1^r, p_2^s, g_1^r, g_2^s, m_i h^{r+s}, \ldots)$ for some $r, s \in \mathbb{Z}_p$ and $m_i \in \mathbb{G}$. Since the simulator constructed g_1, g_2 such that $g_1^x = g_2^y = h$ then it necessarily holds that $(p_1^{ra} p_2^{sb}) = (g_1^{rx} g_2^{sy}) = h^{r+s}$. Let us consider an honest **R** that requests index i from **S**. It selects $v_1, v_2 \in \mathbb{Z}_p$ and sets $d_1 = p_1^r p_1^{v_1}, d_2 = p_2^s p_2^{v_2}$, sending request $Q = (d_1, d_2, \pi)$. Let $R = (s, \delta)$ be the response from **S**. If PoK δ verifies, then (by the soundness property of the proof system) with all but negligible probability $s = d_1^a d_2^b$ and the honest **R** computes the message as $(m_i h^{s+r})/(d_1^a d_2^b h^{-v_1} h^{-v_2}) = m_i$. This is identical to the decryption obtained by $\mathcal{S}$ using the trapdoor (x, y), which produces $h^{s+r}/(g_1^{rx} g_2^{sy}) = m_i$. Thus, the distribution of messages given to $\mathcal{F}_{OT}^{N\times 1}$ by $\mathcal{S}$ is indistinguishable from the distribution of messages obtained by running the protocol directly with **S**.

$\square$

Claim A.3 *When $\mathcal{A}$ corrupts neither* **S** *nor* **R**, *then* $\mathsf{IDEAL}_{\mathcal{F}_{OT}^{N\times 1}, \mathcal{S}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA}, \mathcal{A}, \mathcal{Z}}$ *under the DLIN and N-*Strong DLIN *assumptions.*

We omit a formal proof of this claim, but note that it re-uses techniques identical to those of the previous claims. Specifically, we replace the Sender's initial message T with a commitment to a random database, and show that this random database is indistinguishable from a real database under the *N-Strong DLIN* assumption (as in Claim A.1). We then argue that by the Witness-Indistinguishability property of the Groth-Sahai proof system, the extractions on message index 1 are indistinguishable from extractions on other message indices (as in Claim A.2).

Claim A.4 *When $\mathcal{A}$ corrupts both* **S** *and* **R***, then* $\mathsf{IDEAL}_{\mathcal{F}_{OT}^{N\times 1},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathsf{OTA},\mathcal{A},\mathcal{Z}}$.

We omit a formal proof of this claim.

Lemma A.5 *Under the DLIN assumption, the parameters generated by* $\mathsf{GSSetup}$ *(and* $\mathsf{GSExtractSetup}$*) are computationally indistinguishable from those produced by* $\mathsf{GSSimulateSetup}$.

We omit a proof of this lemma, and refer the reader to the work of Groth and Sahai [23]. Briefly, in the DLIN instantiation, the normal parameters embed a tuple of the form $(g, g^a, g^b, g^{az}, g^{bs}, g^w)$. The normal (and extraction) parameters have $w = z+s$, while the simulation parameters have $w \in_R \mathbb{Z}_p$. If DLIN holds in $\mathbb{G}$, then the two types of parameter are computationally indistinguishable.

Lemma A.6 *Under the* N-HSDH *and CDH*[3] *assumptions, the probability that* **S** *outputs* Extract-Fail *in* **Game 3** *is negligible.*

Proof sketch. Let $T = (pk, C_1, \ldots, C_N)$ be honestly-generated as in **Game 3**. Consider $\mathcal{A}$'s request (sid, Q_i) at transfer $i \in [1, k]$, and parse Q_i as (d_1, d_2, π) where π is a PoK (of the statement described in the definition of $\mathsf{OTRequest}$) using parameters GS'_R. Note that the simulator knows the trapdoor td_{ext} corresponding to GS'_R, and can therefore extract a satisfying witness $W = (\omega_1, \ldots, \omega_{10}) \leftarrow \mathsf{GSExtract}(GS_R, td_{ext}, \pi)$ (in the general case, extraction succeeds with probability $\geq 1 - \nu(\kappa)$ by the Soundess property of the Groth-Sahai proof system).[4] Since extraction fails with at most negligible probability, then if **S** outputs Extract-Fail with non-negligible probability, then it must be that that for $j \in [1, N]$ there is either (a) *no* single ciphertext $C_j = (c_{j,1}, \ldots, c_{j,9})$ such that $(\omega_1, \omega_2) = (c_{j,1}, c_{j,2})$, *or* (b) there are multiple ciphertexts for which the relation holds.

We can easily dispose of case (b): since T is honestly generated, then for each ciphertext $(c_{j,1}, \ldots, c_{j,9})$, the values $c_{j,1}, c_{j,2}$ are uniformly distributed in $\mathbb{G}$. Therefore, the probability is negligible that any two distinct ciphertexts are identical in the first two elements. This it remains only to address case (a) where there is no ciphertext C_j such that $(c_{j,1}, c_{j,2}) = (\omega_1, \omega_2)$. This condition can be further divided into two sub-cases:

1. Where for $i \neq j$ there exists some pair of ciphertexts C_i, C_j such that $\omega_1 = c_{i,1}$ and $\omega_2 = c_{j,2}$.
2. Where there is no pair of ciphertexts such that the above condition holds, *i.e.,* either ω_1 or ω_2 is not contained within any ciphertext in T.

We now show that if $\mathcal{A}$ outputs a PoK satisfying condition (1) then we can use its response to solve the CDH problem, and if $\mathcal{A}$ satisfies condition (2) we can solve N-HSDH. We now describe each of the two simulations:

Case 1: CDH. We consider the case where $\mathcal{A}$ produces $(\omega_1, \omega_2) = (c_{i,1}, c_{j,2})$ for $i \neq j$, and show that an $\mathcal{A}$ that produces such a query can be used to solve the Computational Diffie-Hellman

[3]Computational Diffie-Hellman (CDH) is implied by DLIN ; thus, no new assumptions are being introduced here.
[4]In fact, for parameters $GS'_R \in \mathsf{GSExtractSetup}()$, Groth-Sahai proofs are *perfectly* extractable [23].

problem in $\mathbb{G}$, *i.e.,* given (g, g^a, g^b) for $a, b \in_R \mathbb{Z}_p$, solve for g^{ab}. The intuition behind this argument is that the final two elements c_8, c_9 of each ciphertext represent a signature on the product $(c_1 c_2)$. This signature is built from the Boneh-Boyen selective-ID IBE scheme from [4] (§4), and a forger of this scheme can be used to solve the CDH problem.[5] Our reduction is based on the one given by Boneh and Boyen, although we reduce to CDH. Since the DLIN assumption implies the hardness of CDH, we are not introducing a new assumption.

Given an input (g, g^a, g^b) to the CDH problem, select $t, u, v, w, x_1, x_2, y \xleftarrow{\$} \mathbb{Z}_p$, and set $g_1 \leftarrow g^b$, $g_2 \leftarrow g$, $h \leftarrow (g^a)^{-v} g^w$. Set $\mathsf{crs} \leftarrow (g_1, g_2, h)$, and $pk \leftarrow ((g^a)^t, (g^a)^u, (g^a)^{tx_1}, (g^a)^{ux_2}, g^a)$. Randomly select two ciphertext indices i^*, j^* such that $i^* \neq j^*$.

Now for $i = 1$ to N, choose r_i, s_i, y_i uniformly from $\mathbb{Z}_p$ with the restriction that $(tr_{i^*} + us_{j^*}) = v \bmod p$. Set $z_i = (tr_i + us_i) \bmod p$, and construct the i^{th} ciphertext as:

$$C_i = \left((g^a)^{tr_i}, (g^a)^{us_i}, g_1^{r_i}, g_2^{s_i}, m_j h^{r_i+s_i}, (g^a)^{\frac{t}{x_1+r_i}}, (g^a)^{\frac{u}{x_2+s_i}}, (g^b)^{\frac{-1}{z_i-v}} g^{y_i}, (g^b)^{\frac{-w}{z_i-v}} ((g^a)^{z_i-v} g^w)^{y_i} \right)$$

(Note that the last two elements have the correct distribution. Let $\tilde{y_i} = y_i - b/(z_i - v)$, and re-write $(g^b)^{\frac{-1}{z_i-v}} g^{y_i} = g^{y_i - b/(z_i-v)} = g^{\tilde{y_i}}$. We can then express the final element $(g^b)^{\frac{-w}{z_i-v}}((g^a)^{z_i-v} g^w)^{y_i}$ as $(g^b)^a \left((g^a)^{z_i-v} g^w\right)^{y_i - \frac{-b}{z_i-v}} = (g^{ab}) \left(g^{az_i} h\right)^{\tilde{y_i}} = g_1^a((g^a)^{tr_i+us_i} h)^{\tilde{y_i}}$.)

Now set $T \leftarrow (pk, C_1, \ldots, C_N)$ and send T to $\mathcal{A}$. Whenever $\mathcal{A}$ submits a request $Q = (d_1, d_2, \pi)$ where π verifies correctly, extract $(\omega_1, \ldots, \omega_{10})$ from π. Now:

1. If, for some $j \in [1, N]$, the pair $(\omega_1, \omega_2) = ((g^a)^{tr_j}, (g^a)^{us_j})$: then output a valid response to $\mathcal{A}$ by selecting $s' = (h^{r_j+s_j} \omega_9 \omega_{10})$, constructing the proof δ', and sending $R = (s', \delta')$ to $\mathcal{A}$.[6] Continue the simulation.
2. If $(\omega_1, \omega_2) = ((g^a)^{tr_{i^*}}, (g^a)^{us_{j^*}})$, then output ω_8/ω_7^w as the solution to the CDH problem.
3. In all other cases, abort the simulation.

Observe that in case (2) the soundness of the G-S proof system ensures that for some y' we can represent $(\omega_7, \omega_8) = (g^{y'}, ((g^a)^v h)^{y'} g^{ab})$. By substitution we obtain $(g^{y'}, ((g^a)^v (g^a)^{-v} g^w)^{y'} g^{ab}) = (g^{y'}, g^{wy'} g^{ab})$, and thus $\omega_8/\omega_7^w = g^{ab}$. In this case, we can obtain the value g^{ab} and output a correct solution to the CDH problem.

Note that the distribution of the messages sent to $\mathcal{A}$ is identical to that of the real attack, and are independent of i^*, j^* in $\mathcal{A}$'s view. Therefore, if $\mathcal{A}$ produces $(\omega_1, \omega_2) = (c_{i',1}, c_{j',2})$ for $i' \neq j'$ with some non-negligible probability ϵ, then the approach above solves CDH with probability approximately $\frac{\epsilon}{N^2-N}$, *i.e.,* the probability that that $(i', j') = (i^*, j^*)$.

Case 2: N-HSDH. In the case where either ω_1 or ω_2 is not contained within any ciphertext in T, then we will construct a solver for the N-HSDH problem. Our simulation proceeds as follows: given the N-HSDH instance $(g, g^x, \hat{h}, \{g^{t_1}, \hat{h}^{t_1}, g^{\frac{1}{x+t_1}}\}, \ldots, \{g^{t_N}, \hat{h}^{t_N}, g^{\frac{1}{x+t_N}}\})$, randomly select $s \in \{1, 2\}$ representing one of the following two strategies. (We will use boldfaced type to indicate elements drawn from the HSDH instance.)

[5]Note that a selective-ID IBE scheme implies a secure signature scheme *only* if the message space is polynomial in κ. Since in this case $\mathcal{A}$ succeeds by proving knowledge of a signature on message $(c_{i,1} c_{j,2})$ for some $i \neq j$, we have naturally restricted the total number of messages to $N^2 - N$.

[6]Note that we can simulate the proof δ', but this is not even necessary, since $(h^{r_j} \omega_9, h^{s_j} \omega_{10}, h)$ is a valid witness to the statement.

Strategy 1. Select $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and set $h \leftarrow g^a$, $g_1 \leftarrow \hat{h}$ and $g_2 \leftarrow \hat{h}^c$. Embed (g_1, g_2, h) into crs. Select p_1, p_2 such that $p_1 = g, p_2 = h^{1/b}$. Thus $sk = (a, b)$. Select $y, \alpha \xleftarrow{\$} \mathbb{Z}_p$ and set $p_3 = g^x, p_4 = p_2^y, p_5 = g_2^\alpha$. To compute each ciphertext C_j, select $s_j, z_i \in \mathbb{Z}_p$ and compute $C_j = \left(\mathbf{g}^{\mathbf{t_j}}, p_2^{s_j}, \hat{\mathbf{h}}^{\mathbf{t_j}}, g_2^{s_j}, \mathbf{g}^{\mathbf{t_j}a} h^{s_j} m_j, \mathbf{g}^{\frac{\mathbf{1}}{\mathbf{x+t_j}}}, p_2^{\frac{1}{y+s_j}}, g_2^{z_j}, (\mathbf{g}^{\mathbf{t_j}} p_2^{s_j} h)^{z_j} g_1^\alpha\right)$.

Strategy 2. Select $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and set $h \leftarrow g^a$, $g_1 \leftarrow \hat{h}^c$ and $g_2 \leftarrow \hat{h}$. Embed (g_1, g_2, h) into crs. Select p_1, p_2 such that $p_1 = h^{1/b}, p_2 = g$. Thus $sk = (b, a)$. Select $y, \alpha \xleftarrow{\$} \mathbb{Z}_p$ and set $p_3 = p_1^y, p_4 = g^x, p_5 = g_2^\alpha$. To compute each ciphertext C_j, select $s_j, z_j \in \mathbb{Z}_p$ and compute $C_j = \left(p_1^{s_j}, \mathbf{g}^{\mathbf{t_j}}, g_1^{s_j}, \hat{\mathbf{h}}^{\mathbf{t_j}}, h^{s_j} \mathbf{g}^{\mathbf{t_j}a} m_j, p_1^{\frac{1}{y+s_j}}, \mathbf{g}^{\frac{\mathbf{1}}{\mathbf{x+t_j}}}, g_2^{z_j}, (p_1^{s_j} \mathbf{g}^{\mathbf{t_j}} h)^{z_j} g_1^\alpha\right)$.

In both cases, set $pk = (p_1, \ldots, p_5)$ and $T = (pk, C_1, \ldots, C_N)$. Observe that since $t_1, \ldots, t_N$ are uniformly distributed, then T has the correct distribution. Answer $\mathcal{A}$'s queries using the key sk. If ever $\mathcal{A}$ outputs a PoK π such that for Strategy $s \in \{1, 2\}$ the extracted witness ω_s does not match any $c_{j,s} \in C_j$, then output $(\omega_s, \omega_{s+2}, \omega_{s+4})$ which— by the soundness property of the proof system— can be expressed as $(g^{t'}, \hat{h}^{t'}, g^{1/(x+t')})$ for some $t' \notin \{t_1, \ldots, t_N\}$. Otherwise abort. This tuple represents a valid solution to the HSDH problem. Since all values are correctly distributed and s is outside of $\mathcal{A}$'s view, then we select the correct strategy with probability 1/2.

To conclude our sketch, note that we have covered all cases where event Extract-Fail can occur. Thus if the event occurs with probability non-negligible in κ then we have an algorithm that solves N-HSDH or CDH with non-negligible probability.

□

Lemma A.7 *Replacing* **S***'s honestly-generated responses (as in* **Game 2***) with simulated responses (as in* **Game 3***) results in a simulation that is computationally indistinguishable from that of* **Game 2**.

Proof sketch. Consider a transcript where each response (sid, R) is replaced with a simulated response (sid, R'). Let $R = (s, \delta)$ be the honestly-generated response, and let $R' = (s', \delta')$ be the simulated response. To complete our argument, we must show that for any given reponse: (1) with probability at most $\nu(\kappa)$, the value $s \neq s'$, and (2) the PoK $\delta \overset{c}{\approx} \delta'$. This must hold for all $\mathcal{A}, \mathcal{Z}$.

Recall that pk embeds $p_1^{x_1} = p_2^{x_2} = h$ for some $x_1, x_2 \in \mathbb{Z}_p$. $\mathcal{A}$ initiates the transfer by sending a message (sid, Q) containing the values (d_1, d_2, π). Using the extraction algorithm, we obtain a witness $W = (\omega_1, \ldots, \omega_{10})$ to the statement S_π. Note that a correctly-formed response will have the form $s = (d_1^{x_1} d_2^{x_2})$, and for $C_{\sigma'} = (c_1, \ldots, c_9)$ a simulated response has the form $s' = (c_5 \omega_9 \omega_{10})/m_{\sigma'}$, which we expand to $s' = (c_1^{x_1} c_2^{x_2} m_{\sigma'} h^{v_1} h^{v_2})/m_{\sigma'}$ for some (v_1, v_2). We omit a detailed expansion, but observe that by the statement S_π it holds that $d_1 = c_1 h^{v_1/x_1}$ and $d_2 = c_2 h^{v_2/x_2}$ and thus our simulated s' is identical to the correct response s.

Paraphrasing the composable zero-knowledge property of the Groth-Sahai proof system, when $(GS'_S, td_{sim}) \in \mathsf{GSSimulateSetup}()$ we can simulate a PoK $\delta' \leftarrow \mathsf{GSSimProve}(GS'_S, S_\delta)$ such that no adversary can distinguish δ' from a valid PoK. It remains to show that we can simulate on statement S_δ. Recall that the δ is defined as:

$$\delta = NIZK_{GS_S}\{(a_1, a_2, a_3) : \\ e(a_1, p_1)/e(d_1, a_3) = 1 \;\wedge\; e(a_2, p_2)/e(d_2, a_3) = 1 \;\wedge \\ e(s, a_3)/e(a_1 \cdot a_2, h) = 1 \;\wedge\; e(g, a_3)/e(g, h) = 1\}$$

To simulate the proof, we must select commitments $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$ to represent a_1, a_2, a_3, and we then compute opening values such that each statement is satisfied. Note that using the simulation trapdoor td_{sim} we may open the commitment differently in each statement. To simulate a proof δ', set $a_1 = a_2 = a_3 = h^0$ and generate commitments to each value. In the first three statements, we open $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$ correctly to h^0. In the final statement, we use the simulation trapdoor to open the commitment $\tilde{c}_3$ to h^1. Thus, all statements are satisfied.

□

Lemma A.8 *Let $m_1, \ldots, m_N \in \mathbb{G}$ be any message database, and $\hat{m}_1, \ldots, \hat{m}_N \in_R \mathbb{G}$ be a set of random messages. Also let all of* **S***'s responses be computed as in* **Game 3***. Under the N-Strong Decision Linear assumption, no environment $\mathcal{Z}$ will distinguish the transcript where $T =$* $\mathsf{OTInitialize}(\mathsf{crs}, m_1, \ldots, m_N)$ *from the transcript where $T =$* $\mathsf{OTInitialize}(\mathsf{crs}, \hat{m}_1, \ldots, \hat{m}_N)$ *(except with negligible probability).*

Proof sketch. Let $D = (\hat{u}, \hat{v}, \hat{h}, \hat{u}^{x_1}, \hat{v}^{x_2}, \{u^{r_i}, v^{s_i}, u^{1/(x_1+r_i)}, v^{1/(x_2+s_i)}, Q_i\}_{i\in[1,N]})$ be a candidate Strong Decision Linear tuple. Next, consider a simulation which behaves as follows:

1. Select random $z_1, z_2 \in \mathbb{Z}_p$, set $g_1 = \hat{u}^{z_1}, g_2 = \hat{v}^{z_2}, h = \hat{h}$ and fix $\mathsf{crs} \leftarrow (\gamma, GS'_S, GS'_R, g_1, g_2, h)$.
2. Set $pk = (p_1, \ldots, p_5)$ as: $p_1 \leftarrow \hat{u}, p_2 \leftarrow \hat{v}$, $p_3 \leftarrow \hat{u}^{x_1}, p_4 \leftarrow \hat{v}^{x_2}, p_4 \leftarrow g_2^{x_3}$, for random $x_3 \in \mathbb{Z}_p$.
3. For $i = 1$ to N, choose a fresh random $t \in \mathbb{Z}_p$ and set the ciphertext value as:

$$C_i = (\hat{u}^{r_i}, \hat{v}^{s_i}, \hat{u}^{r_i z_1}, \hat{v}^{s_i z_2}, Q_i \cdot X_i, \hat{u}^{1/(x_1+r_i)}, \hat{v}^{1/(x_2+s_i)}, g_2^t, (\hat{u}^{r_i}\hat{v}^{s_i}h)^t g_1^{x_3}),$$

   where the X_i values are either all from $(m_1, \ldots, m_N)$ or all from $(\hat{m}_1, \ldots, \hat{m}_N)$.
4. Set $T \leftarrow (pk, C_1, \ldots, C_N)$.
5. The simulation proceeds as in **Game 3** at answer transfer requests.

Let us call the case where all $X_i = m_i$ case one (as in **Game 3**) and likewise when $X_i = \hat{m}_i$ case two (as in **Game 4**). Now, suppose for the sake of contradiction, that there exists a $\mathcal{Z}$ who can distinguish case one from case two with non-negligible probability ϵ. Then, we can use $\mathcal{Z}$ to decide Strong Decision Linear. In the above, if $Q_i = h^{r_i+s_i}$ for each i, then the above simulation perfectly encrypts $(m_1, \ldots, m_N)$. However, when the Q_i values are random elements of $\mathbb{G}$, then the corresponding encrypted messages are random elements in $\mathbb{G}$. □

□

B Double-Trapdoor BBS Encryption

In Section 4, we make sure of a ciphertext which is based on the prior work of Boneh, Boyen, and Shacham [5] and has some interesting properties, which may be useful elsewhere.

Boneh, Boyen and Shacham [5] describe a semantically-secure encryption scheme based on the Decision Linear (DLIN) assumption. We extend their scheme into a *two-key* (double-trapdoor) encryption scheme with a public consistency check. In this system, we can encrypt a message under two distinct public keys pk_1, pk_2, such that *either* of the corresponding secret keys sk_1, sk_2 will decrypt the ciphertext. For every well-formed ciphertext, it must be the case that decryption will produce the same message regardless of which secret key is used. To complete our list of requirements, we also define a *publicly-computable* check for ciphertext well-formedness (*i.e.,* the check does not require knowledge of either secret key).

Define a global parameter $h \in \mathbb{G}$, and let each public key be the random pair $u, v \in \mathbb{G}$, with the secret key $x, y \in \mathbb{Z}_p$ such that $u^x = v^y = h$. To encrypt a message $m \in \mathbb{G}$ under $pk_1 = (u_1, v_1), pk_2 = (u_2, v_2)$, first select random values $r, s \in \mathbb{Z}_p$ and output the ciphertext $(u_1^r, u_2^r, v_1^s, v_2^s, h^{r+s} \cdot m)$. To decrypt a message $(c_1, \ldots, c_5)$ under $sk_1 = (x_1, y_1)$, output $c_5/(c_1^{x_1} \cdot c_3^{y_1})$. To decrypt under $sk_2 = (x_2, y_2)$, output $c_5/(c_2^{x_2} \cdot c_4^{y_2})$. Note that the structure of a ciphertext can be verified using the bilinear map, by checking that (u_1, c_1, u_2, c_2) and (v_1, c_3, v_2, c_4) are DDH tuples.

It is fairly straightforward to show that this extension is also semantically-secure under the DLIN assumption.

C Groth/Sahai Commitments in the DLIN setting

This appendix provides a brief description of the Groth-Sahai commitment scheme. For a more detailed explanation covering the entire proof system, the reader should refer to the original work [23], or to the summary of [3] from which we draw notation. Groth-Sahai commitments can be instantiated in three settings: asymmetric bilinear groups where the SXDH assumption holds [30, 5, 2], composite-order groups where the subgroup decision assumption holds [7], and symmetric groups where DLIN holds [5]. This explanation will be restricted to the DLIN instantiation, since that is the instantiation used in this work.

Groth-Sahai proofs are based on a homomorphic commitment scheme that can be instantiated in one of two modes. In the first approach, which leads to a perfectly-*binding* scheme, the commitment parameters embed a DLIN tuple of the form $(g^a, g^b, g, g^{as}, g^{bz}, g^w)$ for random a, b, s, z and $w = s+z$. In the second approach, which leads to a perfectly-*hiding* scheme, the value w is selected uniformly at random from $\mathbb{Z}_p$. Note that the two distributions are computationally indistinguishable under the DLIN assumption.

To commit to a value $m \in \mathbb{G}$, select random $x_1, x_2, x_3 \in \mathbb{Z}_p$ and compute $(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3) = (g^{ax_1} g^{asx_3}, g^{bx_2} g^{bzx_3}, m g^{x_1+x_2} g^{wx_3})$. Note that for $w = s + z$ this is equivalent to a BBS encryption of m. Opening the commitment involves revealing $(m; x_1, x_2, x_3)$. Additionally, in this case a simulator that knows the extraction trapdoor information (a, b) can decrypt any commitment to reveal the committed value m. In the second instantiation, where w is uniformly distributed in $\mathbb{Z}_p$, the commitment perfectly hides m. However, note that a simulator with knowledge of the trapdoor (a, b, s, z) can select $h \in \mathbb{G}$ and generate a random commitment $(h^{\alpha_1}, h^{\alpha_2}, h^{\alpha_3})$. Then for any $y \in \mathbb{Z}_p$ the simulator can solve for the opening $(h^y; x'_1, x'_2, x'_3)$ by solving the equations $\alpha_1 = ax'_1 + asx'_3, c_2 = bx'_2 + bzx'_3$ and $c_3 = y + x'_1 + x'_2 + (s+z)x'_3$. Note that the Groth-Sahai commitment scheme retains the (multiplicatively) homomorphic properties of BBS encryption. Let $\tilde{c}_i$ be a commitment to message m_1, $\tilde{c}_2$ to message m_2. Then $\tilde{c}_1 \cdot \tilde{c}_2$ (defined as entry-wise multiplication of the elements of $\tilde{c}_1, \tilde{c}_2$) is a commitment to $m_1 m_2$.